

1  
2 **CLAIMS:**

3 1. A kernel emulator for non-native program modules, the emulator  
4 comprising:

5 an interceptor configured to intercept kernel calls from non-native program  
6 modules;

7 a call-converter configured to convert non-native kernel calls intercepted by  
8 the interceptor into native kernel calls.  
9

10 2. An emulator as recited in claim 1, wherein the call-converter  
11 comprises a translator configured to translate a non-native paradigm for passing  
12 parameters into a native paradigm for passing parameters.  
13

14 3. An emulator as recited in claim 1, wherein the call-converter  
15 comprises a translator configured to translate non-native CPU instructions into  
16 native CPU instructions.  
17

18 4. An emulator as recited in claim 1, wherein the call-converter  
19 comprises a translator configured to translate addresses from non-native length  
20 into native length.  
21

22 5. An emulator as recited in claim 1, wherein the call-converter  
23 comprises an argument-converter configured to convert non-native argument  
24 format into native argument format.  
25

1           6.    An emulator as recited in claim 1, wherein the call-converter  
2 comprises a translator configured to translate words from non-native word size  
3 into native word size.

4  
5           7.    An emulator as recited in claim 1 further comprising a memory  
6 constrainer configured to limit addressable memory to a range addressable by non-  
7 native program modules.

8  
9           8.    An emulator as recited in claim 1 further comprising a shared-  
10 memory manager configured to manage memory space that is accessible to both  
11 native and non-native program modules.

12  
13           9.    An emulator as recited in claim 1 further comprising a shared-  
14 memory manager configured to synchronize a native shared data structure with a  
15 non-native shared data structure.

16  
17           10.   An emulator as recited in claim 1 further comprising a shared-  
18 memory manager configured to manage memory space that is accessible to both  
19 native and non-native program modules, wherein the shared-memory manager  
20 maps versions of process shared data structures (SDSs) and versions of thread  
21 shared data structures (SDSs) between native and non-native program modules.

22  
23           11.   An operating system on a computer-readable medium, comprising:  
24 a native kernel configured to receive calls from native program modules;  
25

1 a kernel emulator as recited in claim 1 configured to receive calls from non-  
2 native program modules.

3  
4 **12.** An operating system on a computer-readable medium, comprising:  
5 a native kernel configured to receive calls from native APIs;  
6 a kernel emulator as recited in claim 1 configured to receive calls from non-  
7 native APIs.

8  
9 **13.** A method of emulating a kernel for non-native program modules,  
10 the method comprising:  
11 intercepting kernel calls from non-native program modules;  
12 converting the intercepted non-native kernel calls into native kernel calls.

13  
14 **14.** A method as recited in claim 13, wherein the converting step  
15 comprises translating a non-native paradigm for passing parameters into a native  
16 paradigm for passing parameters.

17  
18 **15.** A method as recited in claim 13, wherein the converting step  
19 comprises translating non-native CPU instructions into native CPU instructions.

20  
21 **16.** A method as recited in claim 13, wherein the converting step  
22 comprises translating addresses from non-native length into native length.

1           **17.** A method as recited in claim 13, wherein the converting step  
2 comprises translating words from non-native word size into native word size.

3  
4           **18.** A method as recited in claim 13 further comprising limiting  
5 addressable memory to a range addressable by non-native program modules.

6  
7           **19.** A method as recited in claim 13 further comprising synchronizing a  
8 native shared data structure with a non-native shared data structure.

9  
10           **20.** A method as recited in claim 13 further comprising mapping  
11 versions of process shared data structures (SDSs) between native and non-native  
12 program modules.

13  
14           **21.** A method as recited in claim 19, wherein a process SDS of a native  
15 program module includes a pointer to a process SDS of a non-native program  
16 module.

17  
18           **22.** A method as recited in claim 19, wherein a process SDS of a non-  
19 native program module includes a pointer to a process SDS of a native program  
20 module.

21  
22           **23.** A method as recited in claim 13 further comprising mapping  
23 versions of thread shared data structures (SDSs) data structure between native and  
24 non-native program modules.

1           **24.**    A method as recited in claim 22, wherein a thread SDS of a native  
2 program module includes a pointer to a thread SDS of a non-native program  
3 module.

4  
5           **25.**    A method as recited in claim 22, wherein a thread SDS of a non-  
6 native program module includes a pointer to a thread SDS of a native program  
7 module.

8  
9           **26.**    A computer comprising one or more computer-readable media  
10 having computer-executable instructions that, when executed by the computer,  
11 perform the method as recited in claim 13.

12  
13           **27.**    A computer-readable medium having computer-executable  
14 instructions that, when executed by a computer, performs the method as recited in  
15 claim 13.

16  
17           **28.**    An operating system embodied on a computer-readable medium  
18 having computer-executable instructions that, when executed by a computer,  
19 performs the method as recited in claim 13.  
20  
21  
22  
23  
24  
25

1       **29.**    A method comprising:

2       determining whether an initiating program module is a native or non-native;  
3       if the initiating program is non-native:

4           limiting available memory to a range that is addressable by the non-  
5       native program module;

6           establishing non-native a version of a shared memory data structure  
7       that may be synchronized with a native version of the same shared memory  
8       data structure.

9  
10       **30.**   A method as recited in claim 29 further comprising:

11       intercepting kernel calls from the non-native program module;  
12       converting the intercepted non-native kernel calls into native kernel calls.

13  
14       **31.**   A method as recited in claim 29 further comprising emulating a non-  
15       native kernel for which kernel calls from the non-native program module are  
16       intended.

1           **32.**   A computer comprising one or more computer-readable media  
2   having computer-executable instructions that, when executed by the computer,  
3   perform the method as recited in claim 29.

4  
5           **33.**   A computer-readable medium having computer-executable  
6   instructions that, when executed by a computer, performs the method as recited in  
7   claim 29.

8  
9           **34.**   A method comprising emulating a non-native kernel for a native  
10   computing platform so that kernel calls from non-native applications are translated  
11   into calls to a native kernel.

12  
13           **35.**   A method as recited in claim 34, wherein the emulating step  
14   comprises:  
15       translating non-native CPU instructions into native CPU instructions;  
16       translating addresses from non-native length into native length;  
17       limiting addressable memory to a range addressable by non-native program  
18   modules.

19  
20           **36.**   A method as recited in claim 35, wherein the emulating step further  
21   comprises translating a non-native paradigm for passing parameters into a native  
22   paradigm for passing parameters.

1           **37.**    A method as recited in claim 34, wherein the converting step further  
2 comprises translating words from non-native word size into native word size.  
3

4           **38.**    A computer comprising one or more computer-readable media  
5 having computer-executable instructions that, when executed by the computer,  
6 perform the method as recited in claim 34.  
7

8           **39.**    A computer-readable medium having computer-executable  
9 instructions that, when executed by a computer, performs the method as recited in  
10 claim 34.  
11

12           **40.**    A kernel emulator configured to emulate a non-native kernel for a  
13 native computing platform so that kernel calls from non-native applications are  
14 translated into calls to a native kernel.  
15

16           **41.**    An emulator as recited in claim 40, wherein the emulator comprises:  
17           an instruction-translator configured to translate non-native CPU  
18 instructions into native CPU instructions;  
19           an address-translator configured to translate addresses from non-native  
20 length into native length;  
21           an memory constrainer configured to limit addressable memory to a range  
22 addressable by non-native program modules.  
23  
24  
25



42. An operating system on a computer-readable medium, comprising:  
a native kernel configured to receive calls from native program modules;  
a kernel emulator as recited in claim 40 configured to receive calls from  
non-native program modules.

43. A kernel emulator for non-native program modules, the emulator  
comprising:

target-platform determiner configured to determine a target platform of a  
non-native program module, wherein the target-platform determiner comprises:

an instruction-type detector configured to determine the type of non-  
native instructions that the non-native program module employs;

a translator selector configured to select a translator capable of  
translating the non-native instructions determined by the instruction-type  
detector into native instructions; and

at least one translator, which may be selected by the selector,  
configured to translate non-native instructions of the non-native program  
module into native instructions;

a target-platform simulator configured to simulate the selected target  
platform so that calls kernel calls from non-native program modules are converted  
into native kernel calls.

44. An operating system on a computer-readable medium, comprising:  
a native kernel configured to receive calls from native program modules;  
a kernel emulator as recited in claim 43 configured to receive calls from  
non-native program modules.

1  
2       **45.**   A kernel emulator for non-native program modules, the emulator  
3 comprising:

4       an interceptor configured to intercept kernel calls from non-native program  
5 modules;

6       a call-converter configured to convert non-native kernel calls intercepted by  
7 the interceptor into native kernel calls, wherein the call-converter comprises:

8           an instruction-translator configured to translate non-native CPU  
9 instructions into native CPU instructions;

10          an address-translator configured to translate addresses from non-  
11 native length into native length.

12  
13       **46.**   An operating system on a computer-readable medium, comprising:  
14 a native kernel configured to receive calls from native program modules;  
15 a kernel emulator as recited in claim 45 configured to receive calls from  
16 non-native program modules.